Blatt 7 - Gruppe: G1-07

Mike Lenz, Jonas Tesfamariam

18. Juni 2023

Aufgabe 1

a)

Löschungen:
Klinik: (4,'Dummy-Klinik')

b)

Löschungen:
Klinik: (2,'Alpecinklinik Hamburg'); Chirurg: (2, 'Dr. Klenk', 2); Patient: (2, 'John Doe', 2); Behandlung: (2,2);

c)

Löschungen:
Klinik: (3,'Klinik G-Funk');
Abänderungen:
Patient: (3, 'Jane Smith', NULL);

d)

Löschungen:
Klinik: (1,'Backhaus Oetker');
Abänderungen:
Chirurg: (1, 'Dr. Oetker', NULL) (3, 'Dr. Dre', NULL); Patient: (1, 'Max Mustermann', NULL);

e)

Löschungen:
Klinik: (1, 'Backhaus Oetker');
Chirurg: (1, 'Dr. Oetker', 1) (3, 'Dr. Dre', 1);
Patient: (1, 'Max Mustermann', 1); Behandlung: (1,1) (1,3);

f)

Löschungen:
Klinik: (1, 'Backhaus Oetker');
Chirurg: (1, 'Dr. Oetker', 1) (3, 'Dr. Dre', 1);
Patient: (1, 'Max Mustermann', 1); Behandlung: (1,1); Abänderungen:
Behandlung: (1, NULL) - War vorher Eintrag (1,3)

g)

Löschungen:
Klinik: (1, 'Backhaus Oetker');
Chirurg: (1, 'Dr. Oetker', 1) (3, 'Dr. Dre', 1);
Patient: (1, 'Max Mustermann', 1); Behandlung: (1,1) (1,3);

h)

Löschungen:
Klinik: (3, 'Klinik G-Funk');
Patient: (3, 'Jane Smith', 3);
Abänderungen:
Behandlung: (1, NULL) - War vorher Eintrag (1,3);

i)

Wird nicht durchgeführt, da ref(D) DELETE RESTRICT ist und Patient Jane Smith gelöscht wird (wegen DELETE CASCADE). Es existiert jedoch der Eintrag (1,3) in Behandlung, weshalb es nicht funktioniert.

j)

Es können reihenfolgenahängige Ergebnisse auftreten.
Fall 1: Klinik -> Chirurg -> Behandlung -> Patient

Löschungen:
Klinik: (1, 'Backhaus Oetker');
Chirurg: (1, 'Dr. Oetker', 1) (3, 'Dr. Dre', 1);
Behandlung: (1,1) (1,3);
Patient: (1, 'Max Mustermann', 1);

Fall 2: Klinik -> Patient -> Behandlung -> Chirurg
Schlägt fehl, da Max Mustermann gelöscht wird, jedoch in Behandlung noch ein Eintrag existiert, welcher Max Mustermann referenziert.

Aufgabe 2

```
CREATE OR REPLACE FUNCTION vor() RETURNS TRIGGER AS $$
BEGIN
    NEW.pnr := TG_ARGV[0] || NEW.pnr;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;
CREATE TRIGGER vorp
    BEFORE INSERT ON personal
    FOR EACH ROW
    EXECUTE PROCEDURE vor ('P');
CREATE TRIGGER vorm
    BEFORE INSERT ON manager
    FOR EACH ROW
    EXECUTE PROCEDURE vor ('M');

CREATE OR REPLACE FUNCTION cd() RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM mitarbeit m WHERE m.pnr = OLD.pnr;
    RETURN NULL;
END
$$ LANGUAGE plpgsql;
CREATE TRIGGER cd
AFTER DELETE ON personal
FOR EACH ROW
EXECUTE PROCEDURE cd();
CREATE TRIGGER cd
AFTER DELETE ON manager
FOR EACH ROW
EXECUTE PROCEDURE cd();
```

Aufgabe 3

```
CREATE OR REPLACE FUNCTION checkVoraussetzung ()
    RETURNS TRIGGER
    AS $$
BEGIN
    IF EXISTS (
        SELECT
            *
        FROM
            hoeren h
        WHERE
            h.matrnr = new.matrnr
            AND h.vorlnr = new.vorlnr) THEN
        RETURN NEW;
    END IF;
    ABORT;
END
$$
LANGUAGE plpgsql;
CREATE TRIGGER cv
    BEFORE INSERT OR UPDATE ON pruefen
    FOR EACH ROW
    EXECUTE PROCEDURE checkVoraussetzung();

CREATE OR REPLACE FUNCTION delstud ()
    RETURNS void
    AS $$
DECLARE
    mnr INT;
BEGIN
    FOR mnr IN( SELECT DISTINCT
            failed.matrnr FROM (
                SELECT
                    s.matrnr,
          p.vorlnr,
          count(*) AS anz
        FROM
          studenten s, pruefen p
                WHERE
```

```
                    s.matrnr = p.matrnr
                    AND p.note > 4
                GROUP BY
                    s.matrnr, p.vorlnr) failed
            WHERE
                failed.anz > 2)
    LOOP
        DELETE FROM hoeren h where h.matrnr = mnr;
        DELETE FROM pruefen p where p.matrnr = mnr;
        DELETE FROM studenten s where s.matrnr = mnr;
    END LOOP;
END
$$
LANGUAGE plpgsql;
```

Aufgabe 4

```
ALTER TABLE pruefen
    ADD startzeitpunkt timestamp,
    ADD endzeitpunkt timestamp;

CREATE OR REPLACE FUNCTION terminVergabe ()
    RETURNS TRIGGER
    AS $$
DECLARE
    spaetesterTermin TIMESTAMP := CURRENT_TIMESTAMP;
    szp TIMESTAMP;
    ezp TIMESTAMP;
    examDuration INT;
BEGIN
    IF NOT new.startzeitpunkt = NULL AND NOT new.
       ↪ endzeitpunkt = NULL THEN
        RETURN new;
    END IF;
    FOR szp, ezp
  IN(
        SELECT
            p.startzeitpunkt, p.endzeitpunkt
    FROM
      pruefen p
        WHERE
            p.matrnr = new.matrnr
            OR p.persnr = new.persnr
            AND p.startzeitpunkt IS NOT NULL
            AND p.endzeitpunkt IS NOT NULL)
    LOOP
        IF szp > spaetesterTermin THEN
            spaetesterTermin := szp;
        END IF;
        IF ezp > spaetesterTermin THEN
            spaetesterTermin := ezp;
        END IF;
    END LOOP;
    examDuration := 6 * new.ects;
  IF examDuration < 15 THEN
```

```
        examDuration := 15;
    ELSIF examDuration > 45 THEN
        examDuration := 45;
    END IF;
  new.startzeitpunkt := spaetesterTermin + interval '1 hour
    ↪ ';
    new.endzeitpunkt := new.startzeitpunkt + examDuration *
      ↪ INTERVAL '1 minute';
    RETURN new;
END
$$
LANGUAGE plpgsql;
CREATE TRIGGER tv
    BEFORE INSERT ON pruefen
    FOR EACH ROW
    EXECUTE PROCEDURE terminVergabe();
```